Class is 170.

Announcements

Matlab Grader homework, Binary graded. 168 (HW1), 166,165 (HW2) has done the homework. (If you have not done HW talk to me/TA!)

Homework 3 due 5 May TODAY
Homework 4 (SVM +Dictionary Learning) due ~24 May, released soon

Jupiter “GPU” homework. Due 10 May

Today:
• Stanford CNN 12, K-means, EM (Bishop 9)
• Play with Tensorflow playground before class [http://playground.tensorflow.org](http://playground.tensorflow.org)
  Solve the spiral problem

Monday
• Stanford CNN 13, Dictionary Learning,
Projects

- **3-4 person groups preferred**
- **Deliverables:** Poster, Report & main code (plus proposal, midterm slide)
- **Topics** your own or chose form suggested topics. Some **physics inspired**.
- **April 26 groups** due to TA. **5 students not signed up 44 Groups formed.** Guidelines is on Piazza
- **May 5 proposal due.** use dropbox Format “Proposal”+groupNumber
  
  https://www.dropbox.com/request/XGqCV0qXm9LBYz7J1msS

- **Wednesday May 22** Midterm slide presentation.
- **Project discussion, 22 May:** We split into 6 sub-classes. The purpose is to make sure your project is on track, good progress and good goals. Each group gives a ~10 min presentation by all members
  
  - (each person talks for ~2 min, ~1 slide)
  - Motivation & background, which data?
  - small Example,
  - final outcome, (focused on method and data)
  - difficulties,

- There are 7 Groups in each sub-class, thus we have 15 min in total/group. And will use the remaining time for discussion.

- **June 5, 5-8pm** poster. Atkinson Hall with Pizza. Upload June ~3
- **Report and code due Saturday 15 June.**
What’s going on inside ConvNets?

Input Image:
3 x 224 x 224

What are the intermediate features looking for?

Class Scores:
1000 numbers

The general expression of a convolution is

\[ g(x, y) = \omega \ast f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s, t) f(x - s, y - t), \]

where \( g(x, y) \) is the filtered image, \( f(x, y) \) is the original image, \( \omega \) is the filter kernel. Every element of the filter kernel is considered by \(-a \leq s \leq a\) and \(-b \leq t \leq b\).

Depending on the element values, a kernel can cause a wide range of effects.

| Identity | \[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{bmatrix}
\] |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image result ( g(x, y) )</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
</tbody>
</table>

| Edge detection | \[
\begin{bmatrix}
1 & 0 & -1 \\
0 & 0 & 0 \\
-1 & 0 & 1
\end{bmatrix}
\] |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image result ( g(x, y) )</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

| Sharpen | \[
\begin{bmatrix}
0 & -1 & 0 \\
-1 & 5 & -1 \\
0 & -1 & 0
\end{bmatrix}
\] |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image result ( g(x, y) )</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
</tbody>
</table>

| Box blur (normalized) | \[
\frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\] |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Image result ( g(x, y) )</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Reverse engineering

First Layer: Visualize Filters

AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7

Similar to human neuron: First observe oriented edges

\[ w^T x \text{ subject to } ||w|| = 1 \]

Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017
Last Layer: Nearest Neighbors

Recall: Nearest neighbors in pixel space

Test image  L2 Nearest neighbors in feature space

4096-dim vector

Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.
Feature Inversion

Given a CNN feature vector for an image, find a new image that:
- Matches the given feature vector
- "looks natural" (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\text{argmin}} \; \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \| \Phi(\mathbf{x}) - \Phi_0 \|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left( (x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \right)^{\frac{\beta}{2}}$$

Feature Inversion

Reconstructing from different layers of VGG-16
Neural Texture Synthesis: Gram Matrix

Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C

Efficient to compute; reshape features from C x H x W to C x HW

then compute G = FFᵀ
Neural Texture Synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer $i$ gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the Gram matrix giving outer product of features:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad \text{(shape } C_i \times C_j)$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices
7. Backprop to get gradient on image
8. Make gradient step on image
9. GOTO 5

Gatys, Ecker, and Bethge, “Texture Synthesis Using Convolutional Neural Networks”, NIPS 2015
Neural Texture Synthesis

Reconstructing texture from higher layers recovers larger features from the input texture.
Neural Style Transfer

Content Image + Style Image = Style Transfer!

Style transfer = Feature reconstruction loss + Gram matrix
Style image

Output image (Start with noise)

Content image

Style Target

Content Target

Loss Network

Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.
K-means and Expectation Maximization

Mike Bianco and Peter Gerstoft
ECE228
5/6/2019
K-means and expectation maximization (EM) can be considered unsupervised learning

- In **supervised learning**, we have desired machine learning (ML) model output or ‘action’ $y$ based on inputs $x$ (features), and model parameters $\theta$
  - Probabilities of the form: $p(y|x, \theta)$
  - Linear regression and classification, support vector machines, etc.

- In **unsupervised learning**, we are interested in discovering useful patterns in the features. This can be for discovering latent data ‘causes’ or significant ‘groups’
  - Probabilities of the form: $p(x|\theta)$
  - Principal components analysis (PCA), K-means, dictionary learning, etc.
Unsupervised learning

Unsupervised machine learning is inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning.

We are not interested in prediction

Supervised learning: all classification and regression.

\[ Y = w^T X \]

Prediction is important.
Supervised learning: least square classifier (binary)

Training set \{ (x^1, y^1), (x^2, y^2), (x^3, y^3) \}
We are given the two classes
(green = 0, red = 1)
Unsupervised learning: how are features best divided?

Just have features \( \{(x_1^1, x_2^1), (x_1^2, x_2^2), (x_1^3, x_2^3)\} \)
K-means

- **Input**: Points \(x_1, \ldots, x_N \in \mathbb{R}^p\); integer \(K\)
- **Output**: “Centers”, or representatives, \(\mu_1, \ldots, \mu_K \in \mathbb{R}^p\)
- Output also \(z_1, \ldots, z_N \in \mathbb{R}^K\)

**Goal**: Minimize average squared distance between points and their nearest representatives:

\[
\text{cost}(\mu_1, \ldots, \mu_K) = \sum_{n=1}^{N} \min_{j} \|x_n - \mu_j\|
\]

The centers carve \(\mathbb{R}^p\) up into \(k\) convex regions: \(\mu_j\)'s region consists of points for which it is the closest center.
K-means

\[ J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| \mathbf{x}_n - \mu_k \|^2 \]  \hspace{1cm} (9.1)

Solving for \( r_{nk} \)

\[ r_{nk} = \begin{cases} 
1 & \text{if } k = \arg \min_j \| \mathbf{x}_n - \mu_j \|^2 \\
0 & \text{otherwise.} 
\end{cases} \]  \hspace{1cm} (9.2)

Differentiating for \( \mu_k \)

\[ 2 \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \mu_k) = 0 \]  \hspace{1cm} (9.3)

which we can easily solve for \( \mu_k \) to give

\[ \mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}. \]  \hspace{1cm} (9.4)
The two phases of re-assigning data points to clusters and re-computing the clusters are known as the E (expectation) and M (maximization) steps in the context of the K-means algorithm. For this example, we have chosen the value of the objective function $J$ in (9.1) is a quadratic function of $r_{nk}$, and so this result has a simple interpretation, namely set $r_{nk} = 1$ if $k = \arg \min_j \| x_n - \mu_j \|^2$ and $r_{nk} = 0$ otherwise.

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \| x_n - \mu_j \|^2 \\ 0 & \text{otherwise}. \end{cases}$$

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}.$$
Old Faithful, Kmeans from Murphy
Application of K-means to data compression: Vector Quantization

Each pixel $\mathbf{x}_i$ is represented by codebook of $K$ entries $\mu_k$.

Encode($\mathbf{x}_i$) = arg\min_k ||$\mathbf{x}_i - \mu_k$||

Consider $N=64k$ observations, of $D=1$ (b/w) dimension, $C=8$ bit

$N\log_2 K + KC$ bits is needed. $K=4$ gives $128k$ a factor 4.
Mixtures of Gaussians (1)

Old Faithful geyser:
The time between eruptions has a bimodal distribution, with the mean interval being either 65 or 91 minutes, and is dependent on the length of the prior eruption. Within a margin of error of ±10 minutes, Old Faithful will erupt either 65 minutes after an eruption lasting less than $2 \frac{1}{2}$ minutes, or 91 minutes after an eruption lasting more than $2 \frac{1}{2}$ minutes.
Mixtures of Gaussians (2)

Combine simple models into a complex model:

\[ p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k) \]

Component
Mixing coefficient

\[ \forall k : \pi_k \geq 0 \quad \sum_{k=1}^{K} \pi_k = 1 \]
Mixtures of Gaussians (3)
Mixture of Gaussians

• **Mixtures of Gaussians**
  \[
P(x) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k).
\]

• Expressed with latent variable \(z\)
  \[
P(x) = \sum_{z} p(z)p(x \mid z) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)
\]

\[
p(x \mid z_k = 1) = N(x; \mu_k, \Sigma_k)
\]

\[
p(z_k = 1) = \frac{\pi_k}{\sum_{k=1}^{K} \pi_k}
\]

\[
p(x \mid z) = \frac{\sum_{k=1}^{K} \pi_k}{\sum_{k=1}^{K} \pi_k} N(x \mid \mu_k, \Sigma_k)
\]

\[
p(z) = \frac{\sum_{k=1}^{K} \pi_k}{\sum_{k=1}^{K} \pi_k}
\]

\[
p(x, z) = \frac{\sum_{k=1}^{K} \pi_k}{\sum_{k=1}^{K} \pi_k} N(x \mid \mu_k, \Sigma_k)
\]

\[
\forall k : \pi_k \geq 0 \quad \sum_{k=1}^{K} \pi_k = 1
\]
Want to estimate the latent variables for data \( X \)

- Probability of data given latent representation
  
  \[ p(\mathbf{X}|\pi, \mu, \Sigma) = \]

- Log likelihood
  
  \[ \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \]

\[
\ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right\}.
\]
Can’t we just solve for the latent variables by maximimizing log likelihood?

• Log likelihood

\[
\ln p(X \mid \pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right\}.
\]

• Take derivative w.r.t. \( \mu_k \):
Can’t we just solve for the latent variables by maximizing log likelihood?

- Log likelihood
  \[
  \ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}.
  \]

- Take derivative w.r.t. \( \mu_k \):
  
  \[
  0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \Sigma_k (x_n - \mu_k)
  \]

  “responsibility”, from Bayes’s rule:

  \[
  \gamma(z_k) \equiv p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)} = \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}.
  \]

Another quantity that will play an important role is the conditional probability of \( z \) given \( x \). We shall use \( \gamma(z_k) \) to denote \( p(z_k = 1|x) \), whose value can be found using Bayes’ theorem

\[
\gamma(z_k) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)} = \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \Sigma_j)}.
\]
Solving for $\mu_k, \Sigma_k$

Take derivative w.r.t. $\mu_k$:
Solving for $\pi_k$

Use Lagrange multipliers with constraint

$$\sum_{k=1}^{K} \pi_k = 1$$
9. MIXTURE MODELS AND EM

9.3. An Alternative View of EM

An Alternative View of EM

1. Initialize the means $\mu_k$, covariances $\Sigma_k$ and mixing coefficients $\pi_k$, and evaluate the initial value of the log likelihood.

2. **E step.** Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)} \quad (9.23)$$

3. **M step.** Re-estimate the parameters using the current responsibilities

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \quad (9.24)$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \quad (9.25)$$

$$\pi_k^{new} = \frac{N_k}{N} \quad (9.26)$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \quad (9.27)$$

4. Evaluate the log likelihood

$$\ln p(X | \mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k) \right\} \quad (9.28)$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.
Important not to have singularities

Figure 9.7 Illustration of how singularities in the likelihood function arise with mixtures of Gaussians. This should be compared with the case of a single Gaussian shown in Figure 1.14 for which no singularities arise.

If we consider the limit $\sigma_j \to 0$, then we see that this term goes to infinity and so the log likelihood function will also go to infinity. Thus the maximization of the log likelihood function is not a well posed problem because such singularities will always be present and will occur whenever one of the Gaussian components 'collapses' onto a specific data point. Recall that this problem did not arise in the case of a single Gaussian distribution. To understand the difference, note that if a single Gaussian collapses onto a data point it will contribute multiplicative factors to the likelihood function arising from the other data points and these factors will go to zero exponentially fast, giving an overall likelihood that goes to zero rather than infinity. However, once we have (at least) two components in the mixture, one of the components can have a finite variance and therefore assign finite probability to all of the data points while the other component can shrink onto one specific data point and thereby contribute an ever increasing additive value to the log likelihood. This is illustrated in Figure 9.7. These singularities provide another example of the severe over-fitting that can occur in a maximum likelihood approach. We shall see that this difficulty does not occur if we adopt a Bayesian approach. For the moment, however, we simply note that in applying maximum likelihood to Gaussian mixture models we must take steps to avoid finding such pathological solutions and instead seek local maxima of the likelihood function that are well behaved. We can hope to avoid the singularities by using suitable heuristics, for instance by detecting when a Gaussian component is collapsing and resetting its mean to a randomly chosen value while also resetting its covariance to some large value, and then continuing with the optimization.

A further issue in finding maximum likelihood solutions arises from the fact that for any given maximum likelihood solution, a $K$-component mixture will have a total of $K!$ equivalent solutions corresponding to the $K!$ ways of assigning $K$ sets of parameters to $K$ components. In other words, for any given (nondegenerate) point in the space of parameter values there will be a further $K! - 1$ additional points all of which give rise to exactly the same distribution. This problem is known as
Given a joint distribution \( p(X, Z|\theta) \) over observed variables \( X \) and latent variables \( Z \), governed by parameters \( \theta \), the goal is to maximize the likelihood function \( p(X|\theta) \) with respect to \( \theta \).

1. Choose an initial setting for the parameters \( \theta^{\text{old}} \).

2. **E step** Evaluate \( p(Z|X, \theta^{\text{old}}) \).

3. **M step** Evaluate \( \theta^{\text{new}} \) given by

\[
\theta^{\text{new}} = \arg \max_{\theta} Q(\theta, \theta^{\text{old}})
\]  

(9.32)

where

\[
Q(\theta, \theta^{\text{old}}) = \sum_{Z} p(Z|X, \theta^{\text{old}}) \ln p(X, Z|\theta).
\]  

(9.33)

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

\[
\theta^{\text{old}} \leftarrow \theta^{\text{new}}
\]  

(9.34)

and return to step 2.
Gaussian Mixtures

(a) $L = 1$

(b) $L = 2$

(c) $L = 5$

(d) $L = 10$

(e) $L = 20$

(f) $L = 50$
Kmeans and EM (9.3.2)

\[ \Sigma_k = \epsilon I \]

\[ p(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi\epsilon)^{1/2}} \exp \left\{ -\frac{1}{2\epsilon} \|x - \mu_k\|^2 \right\}. \] (9.41)

Whereby the responsibilities

\[ \gamma(z_{nk}) = \frac{\pi_k \exp \left\{ -\|x_n - \mu_k\|^2 / 2\epsilon \right\}}{\sum_j \pi_j \exp \left\{ -\|x_n - \mu_j\|^2 / 2\epsilon \right\}}. \] (9.42)

Becomes delta functions.

And the EM means approach the Kmeans

\[ \mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \] (9.17)